前編

組み込みシステム設計者のための LIN 2.0マイコン実装術

――プロトコル仕様を正しく読みこなすコツ

システムの記事

舘 伸幸

本稿では，ネットワーク・プロトコル規格「LIN（local interconnect network）2.0」について解説する．LINはボディ（車体）系の比較的低速なネットワークを低コストで実現するために策定された規格だが，こうした特徴を家電機器に採用しようという動きがある．本稿では，今回の前編と次回の中編でプロトコルの詳細を，後編でソフトウェアの開発事例を紹介する．今回は，とくに仕様書ではあいまいな部分を中心に，プロトコルについての理解を深める．（編集部）

LIN（local interconnect network）プロトコルは，1999年に最初のバージョン（バージョン1.0）が発表されました．

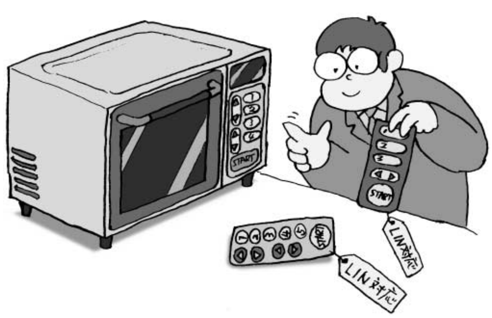


図1　LINの応用分野
冷蔵庫，洗濯機などの家電から自販機類に至るまで，パネル部やメカ制御部，各種センサなどの構成モジュールの接続はLINバスで標準化できると思われる．

2003年にはバージョン2.0（LIN 2.0）の仕様が公開されました．これに伴い，国内でもとくに欧州の自動車メーカ向けに半導体・電装製品が開発されるようになりました．ISO標準化への取り組みも行われています．

また，LINは小規模で，標準化されたネットワーク・プロトコルとして，民生・家電分野への展開も検討されています（**図1**）．車載関連以外の組み込みシステムの開発に携わる方も，一度LINの仕様に目を通してみてはいかがでしょうか．

LINプロトコルの概要についてはこれまでも本誌記事で解説されていますが，今回は実際にマイコンでLINを扱う場合を想定し，その技術的ポイントを解説します．

● 仕様書は七つの項目で構成されている

LINプロトコルは，LINコンソーシアムから配布されているLIN Specification Package Revision 2.0で定義されています．LINプロトコルを実現するためのドライバ・ソフトウェアを作るにせよ，それを使うにせよ，一度は通読しておく必要のあるドキュメントです．LINコンソーシアムのホームページ（http://www.lin-subbus.org/）から，Receive your personal copy of LIN 2.0を選んで，名まえやメール・アドレスを登録することで入手できます．

この仕様書は，**図2**のように七つの項目で構成されています．これらのうち，Specification PackageとProtocol

図2
LIN仕様書の構成
LINの仕様書は七つの項目で構成されている．それぞれの内容については，Specification Packageの2.6章で“DOCUMENT OVERVIEW”として説明されている．Specification Packageに全体の概要が記述されており，残りの六つが仕様書本体となる．かっこ内はその略号．

● Specification Package	全体の概要
● Protocol Specification（PROT）	プロトコル基本仕様
● Diagnostic and configuration Specification（DIAG）	診断の規定
● Physical Layer Specification（PHY）	物理層の規定
● Application Program Interface Specification（API）	APIの規定
● Node Capability Language Specification（NCL）	ノード機能の記述言語仕様
● Configuration Language Specification（CLS）	ノード・コンフィグレーションの記述言語仕様

表1　FEATURES AND POSSIBILITIESの内容

記載されている特性	内容（LIN 1.3との比較）
single master with multiple slaves concept	LIN 1.3からの変更なし．LINのネットワーク・クラスタは，一つのマスタ（親機）に複数のスレーブ（子機）がつながる形態をとる．既存のUARTを利用する小規模で低コストのネットワークを基本としている
low cost silicon implementation based on common UART/SCI interface	
self synchronization without a quartz or ceramics resonator in the slave nodes	ボーレート補正（クラスタの通信レートに対する誤差補正）に関する記述．LIN 1.3ではボーレート補正は±15％だったが，LIN 2.0では±14％に変更されている．また，スレーブはマスタのボーレートを自動検出するようになった
deterministic signal transmission with signal propagation time computable in advance	LIN 1.3では単に“deterministic signal transmission”と記述されていたが，LIN 2.0では「あらかじめ算定できている信号伝播時間で通信を行う」と，タイム・トリガ・プロトコルであることが強調されている
low cost single-wire implementation	LIN 1.3からの変更なし．通常，9,600bps，19,200bpsという通信レートで使うことが多い
speed up to 20 kbit/s	
signal based application interaction	LIN 1.3では明記されていなかった．かといって，とくに新しいしくみが導入されたわけではない

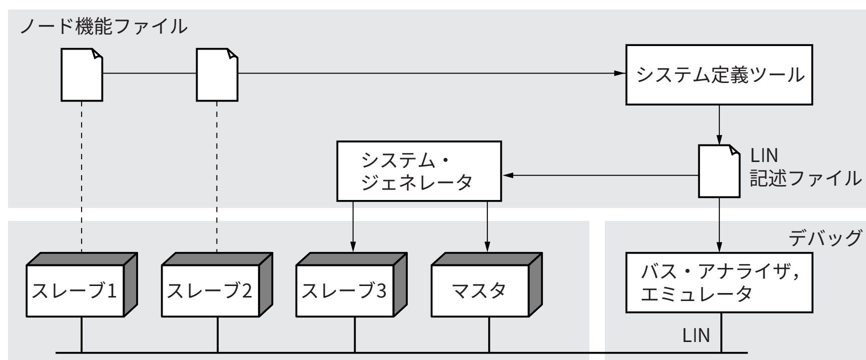


図3
LINクラスタの開発フロー例
スレーブ・ノードはLINクラスタを形成するマスタに接続される．該当するノード機能ファイルは，システム定義プロセスにおいてLIN記述ファイルを生成するシステム定義ツールによって解析される．LIN記述ファイルはシステム・ジェネレータによって解析され，希望するノード（図の例ではマスタおよびスレーブ3）のLIN機能を生成する．また，LIN記述ファイルはバス・アナライザやエミュレータでクラスタ・デバッグの際に使用される．

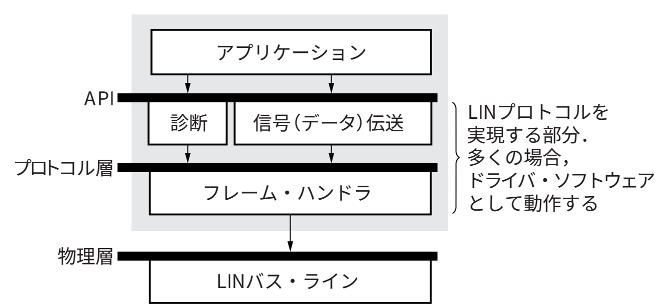


図4　ノード構造
太線が内部的なインターフェースとなる．通常，物理層はUARTやポートである．APIが，ノードのアプリケーションとのインターフェースであり，ソフトウェア資産化のためにも仕様の堅守が重要．

Specificationの二つがLINプロトコルを実装するうえで基本となる仕様です．ここを押さえておけば，残りの部分は，容易に理解を進めることができると思います．

ではSpecification PackageとProtocol Specificationの内容について，順にポイントを押さえながら確認していきましょう．

1. Specification Packageを読み解くポイント

Specification PackageにはLINプロトコルの概要が記述されています．最初のポイントは“FEATURES AND POSSIBILITIES”という章です．この章には，LIN 2.0のおもな特性が記述されています（**表1**）．また，Specification Packageの最後の章で解説されていますが，ここではLINに関連する用語をまず押さえておきましょう．**表2**に用語の意味をまとめておきます．本稿を読み進めるうえで参考にしてください．

次章の“WORK FLOW CONCEPT”では，LINクラスタ開発の際のしくみについて概要が示されています（**図3**）．

“NODE CONCEPT”の章ではソフトウェア構造について述べられています．LIN 1.3の仕様書よりも図がわかりやすくなりました（**図4**）．要は，**図4**のとおりの階層構造のソフトウェアにしなさいということです．API（application program interface）から物理リソース制御（UARTやポート，内部タイマなど，LIN通信に必要となるマイコンのハードウェアの制御）までを，通常，LINドライバと

表2　LIN2.0仕様書で使われる用語

用　語	意　味	関連する仕様番号
active mode	動作状態．cluster（クラスタ）のノードが一つのクラスタとして互いに通信すること	PROT 5
bus interface	クラスタにおける物理的なバス・ワイヤに接続されるノードの回路（トランシーバ，UART など）	—
byte field	LIN バスの各データはバイト・フィールドで送信される．バイト・フィールドには，送信されるスタート・ビット/ストップ・ビットが含まれる	PROT 2.1
checksum model	チェックサム・モデルとしては，標準チェックサム（classic checksum）と拡張チェックサム（enhanced checksum）の二つがある．拡張チェックサムでは保護ID（識別子）も含まれるが，標準チェックサムには含まれていない	PROT 2.1.5
classic checksum	標準チェックサムは，LIN 1.3 までのバージョンおよびdiagnostic frame（診断フレーム）で使用される．データ・バイトのみ加算する	PROT 2.1.5
cluster	クラスタとは，LIN のバス・ワイヤとすべてのノードを指す	—
data	LIN フレームのレスポンス（response）は1～8バイトのデータを伝送するが，これを総称してデータと呼ぶ	PROT 2.1.4
diagnostic frame	診断フレームは，マスタ・リクエスト・フレームとスレーブ・レスポンス・フレームを指す	PROT 2.3.4
enhanced checksum	拡張チェックサムはLIN 2.0 のスレーブ・ノードの通信に使用する．データ・バイトだけでなく，保護ID も含めてチェックサムも計算する	PROT 2.1.5
event triggered frame	イベント・トリガ・フレームは「代替物」として使用され，複数のスレーブ・ノードがレスポンスできるようにする．関連する信号がまれに変更される場合に便利	PROT 2.3.2
frame	すべての情報はフレームで送信される．フレームはヘッダとレスポンスからなる	PROT 2
frame slot	フレーム・スロットとは，LIN バスのフレームを転送するために割り当てられた時間のこと．スケジュール・テーブルにおける一つの入力に相当	PROT 2.2
go-to-sleep command	スリープ要求コマンドは，強制的にスレーブ・ノードをスリープ・モードにするために発信される特別な診断フレーム	PROT 5.2, API 2.5.4
header	ヘッダはフレームのいちばん初めの部分．つねにマスタ・タスクから送信される	PROT 2.1
identifier（ID）	0～63 の範囲のフレームのID（識別子）のこと	PROT 2.1.3
LIN description file（LDF）	LIN 記述ファイルはシステム定義で作成され，開発ツールなどで解析される	CLS，NCL1.1
LIN product identification	LIN 製品ID は，各LIN ノードに割り振られた一意の番号	DIAG 2.4
master node	マスタ・ノードにはスレーブ・タスクだけでなく，バス上の全フレーム・ヘッダを送信するマスタ・タスクも含まれる．バス上のタイミングとスケジュール・テーブルを管理する	—
master request frame	マスタ・リクエスト・フレームのID は60であり，マスタ・ノードによって作成された診断フレームに使用される	PROT 2.3.4, DIAG
master task	マスタ・タスクはバス上の全フレーム・ヘッダを送信する役目を担う．バスのタイミングとスケジュール・テーブルを管理する	PROT 4.1
message identifier	メッセージID は，スレーブ・ノードの各フレームが保有する一意の16 ビットの番号を指す．ノード設定中にこの番号は保護ID に関連づけられ，ノードの通常通信に使用される	DIAG2.5.1
NAD （node address for diagnostic）	診断フレームがブロードキャスト送信されると，NAD はアドレス指定されたスレーブ・ノードを識別する．NAD は物理アドレスおよび論理アドレスである	DIAG2.3.2
node	車載システムにおいて，大まかに言うとノードはECU（electronic control unit；電子制御装置）のことである．しかし，一つのECU は複数のLIN クラスタに接続される場合がある．このときのノードは，「バス・インターフェース」と言い換えるべきだろう	—
node capability file（NCF）	NCF は，LIN バスから見たとおりのスレーブ・ノードを表す．システム定義で使用	NCL1.1
protected identifier	保護ID は，6 ビットのID と2 ビットのパリティからなる1 バイトの値	PROT 2.1.3
publish	信号（もしくは無条件フレーム）は，かならず一つのpublisher を持つ情報源（送信）ノードのこと．subscribe の反対	PROT 2.1.4/4.2
request	マスタ・ノードは，ノード設定および診断トランスポート層でスレーブ・ノードにリクエストを載せる	DIAG 2.31./3.3.1
reserved frame	予約フレームは，使用されないID（63）を持つ	PROT 2.3.6
response	1）LIN フレームはヘッダとレスポンスからなる．「フレーム・レスポンス」とも呼ばれる	PROT 2.1
	2）ISO リクエスト用の応答メッセージのこと．「診断レスポンス」とも呼ばれる	DIAG 2.3.1/3.3.1
schedule table	スケジュール・テーブルは，LIN バス上のトラフィックを識別する	PROT 3, CLS2.5，API2.4
slave node	スレーブ・タスクのみを含むノード（つまり，マスタ・タスクを含まない）	—
slave response frame	スレーブ・レスポンス・フレームのID は61．ある一つのスレーブ・ノードによって作成される診断フレームに使用する	PROT 2.3.4, DIAG
slave task	スレーブ・タスクはバス上のすべてのフレーム・ヘッダをモニタし，それに応じて対処する役目を持つ．つまり，フレーム・レスポンスをpublish（送信）する，もしくはsubscribe（受信）する（あるいは無視する）ということ	PROT 4.2
sleep mode	スリープ・モードは，クラスタでまったく通信がない状態を指す（マイコンでいえば，省電力モードと同じ意味）	PROT 5
signal	信号は，信号送信フレームを使用してLIN クラスタで送信される値を意味する	PROT 1
signal-carrying frame	信号伝送フレームは，0～59 の範囲のID を持つ．無条件フレーム，スポラディック・フレーム，イベント・トリガ・フレームが信号伝送フレームになる	PROT 2.1.3
sporadic frame	スポラディック・フレームは無条件フレームに類似した信号伝送フレームだが，その信号がpublisher によって更新される場合のみ，フレーム・スロットで送信される	PROT 2.3.3
subscribe	subscribe はpublish の反意語で，信号（または信号伝送フレーム）を受信すること	PROT 2.1.4/4.2
system definition	システム定義は，LIN 記述ファイルを作成するプロセス	NCL 1.1.2
system generation	LIN 記述ファイルに対して，クラスタにおいて一つの（またはそれ以上の）ノードをターゲットにするプロセス	NCL 1.1.1
unconditional frame	無条件フレームは，割り当てられたフレーム・スロットにつねに送信される信号伝送フレームを指す	PROT 2.3.1
user-defined frame	ユーザ定義フレームのID は62．用途は仕様でとくに定めていない	PROT 2.3.5

呼ばれるソフトウェアで処理します．多くの場合，半導体メーカやツール・ベンダの設計者がこの部分の開発を担当することになります．一方，セット・メーカの設計者はAPIの仕様を遵守してアプリケーションを作成します．実際には，マイコンの限られたリソースの中で実装しなくてはならないため，いろいろな妥協が必要になります．例えば，速度優先のためにグローバル変数を多用することなどが挙げられます．これには，性能と引き換えにモジュール間やプログラム階層間のインターフェースがあいまいになってしまうという副作用があります．

アプリケーションもドライバも両方開発しなければならない設計者は，APIの境界だけは遵守しておくべきでしょう．**図4**のように，APIはノードの機能を実現するアプリケーションとLINプロトコルを実現するドライバの境界です．アプリケーションの中でハードウェアに依存する記述を行ったり，LINドライバ内部で管理している情報へAPIを介さずに直接アクセスするような構造をとると，移植性や拡張性が大きく損なわれるだけでなく，（ドライバ部分も含めた）ソフトウェア全体の信頼性も低下してしまいます．

“CONCEPT OF OPERATION”の章には，ほぼLIN 1.3と同じ内容が記述されています．**図5**に，LINバスを介したマスタ・ノードとスレーブ・ノードの接続のようすを示します．マスタ・タスクはフレーム・ヘッダを送信するものであり，スレーブ・タスクは（ヘッダを受信して）レスポンスを送信するものです．**図5**に示したようにマスタ・ノードにもスレーブ・タスクは実装されるので，マスタ・ノードのスレーブ・タスクがレスポンスに相当する部分をヘッダに続いて送信することもあります．その場合，指定されたスレーブ・ノードはマスタ・ノードからのデータを受信するだけの動作となります（**図6**）．

この章では，最後に重要な記述があります．マスタ・タスクはスケジュール・テーブルというものに基づいてフレームを伝送するという内容です．すなわち，LIN 2.0はタイム・トリガ・プロトコルであることが明記されています．

なお，一つのLINクラスタは一つのマスタ・ノードと一つ以上のスレーブ・ノードから構成されます．また，LINドライバはマスタ・タスクとスレーブ・タスクから成り立ちます．単に「マスタ」，「スレーブ」と言ったとき，LINではノードのことを指す場合とタスクのことを指す場合があります．混乱しないように注意しましょう．

● LIN 1.3と比べてソフトを開発しやすい仕様になった

“Changes between LIN1.3 and LIN2.0”という章は，タイトルのとおりLIN 1.3とLIN 2.0の違いを列挙しています．その中でもとくに重要と思われる相違点を以下に挙げます．

最初のポイントは，バイト・アレイ信号がサポートされ，

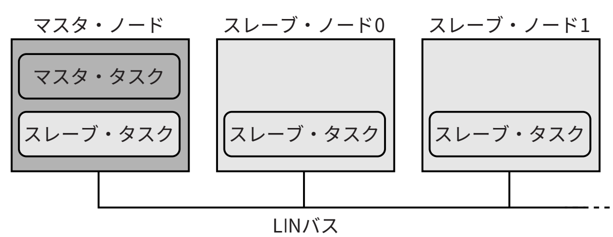


図5　マスタ・タスクとスレーブ・タスク
マスタ・ノードにはマスタ・タスクとスレーブ・タスクを実装する．スレーブ・ノードにはスレーブ・タスクのみ実装する．

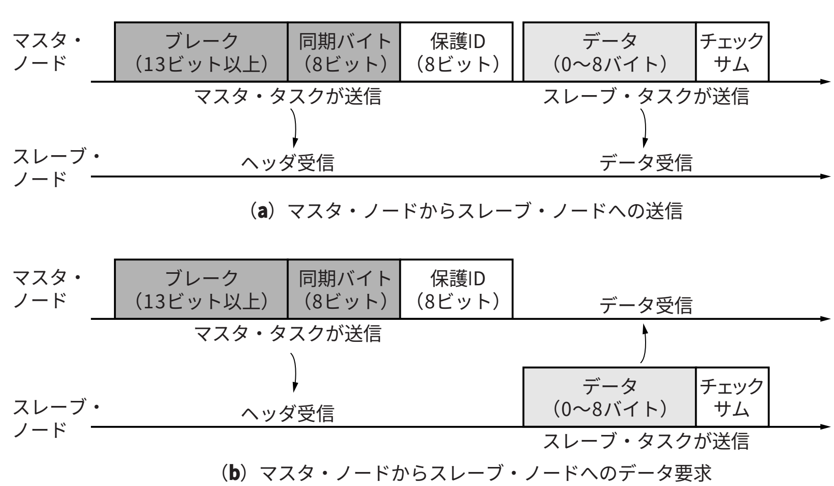


図6
マスタ・スレーブ間通信
マスタ・ノードのスレーブ・タスクがレスポンスに相当する部分をヘッダに続いて送信する場合，(a)のように指定されたスレーブ・ノードはマスタ・ノードからのデータを受信するだけの動作となる．

信号サイズが8バイトまで許容されるようになったという点です．LIN 1.3をご存じの方は，「もともとフレームのデータ域は8バイト分あったのに何が変わったんだ？」と感じるかもしれません．これは，フレーム構造ではなく，単にAPIの仕様が拡張されたことを意味しています．LIN 1.3のAPIでは2バイトまでの読み書き関数しか用意されていませんでした．一方，LIN 2.0では配列（へのポインタ）の読み書きが拡張されています．代わりに，LIN 1.3にあった信号グループという仕様を削除しています．APIを使う側にしてみれば，単にデータ配列を渡せるLIN 2.0のほうが単純で使いやすくなったと思います．

次のポイントは，スポラディック・フレーム（**表2**を参照）が追加された点です．スポラディック（sporadic）とは「散発的な」という意味ですが，スポラディック・フレームとは，変更があまり頻繁ではない信号を扱うためのしくみです（詳細は，本記事の中編で解説予定）．

また，LIN 1.3ではフレームやフィールドの間隔などは「何ビット分」と，ビット単位で規定されていましたが，LIN 2.0では「秒」，すなわち時間単位になりました．プログラムを作る立場としては，こちらのほうが設計しやすくなるので，ありがたい変更です．

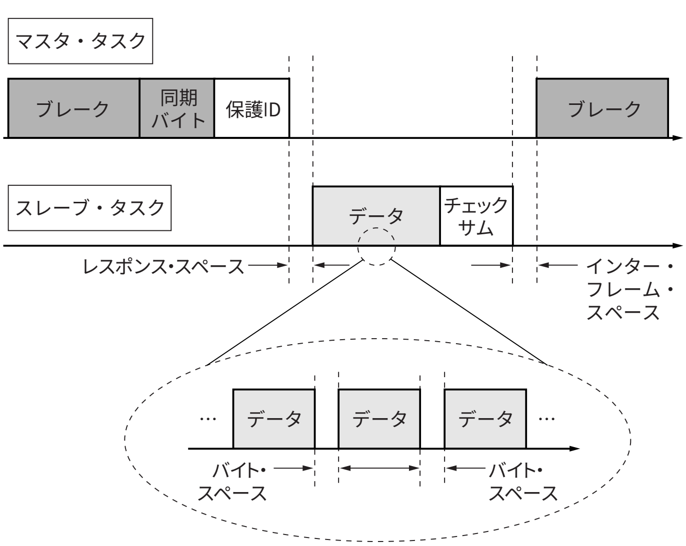


図7　3種類のスペース

ヘッダとレスポンスの間のレスポンス・スペース，フレームとフレームの間のインター・フレーム・スペース，フレームを構成する各バイト間のバイト・スペースの三つのスペースが想定される．

2．プロトコル詳細 ——フレームの構成と種類

次に，本仕様群の中核である“ Protocol Specification（LINプロトコル詳細）”について，重要なポイントを説明します．

● 信号の型はスカラ値かバイト配列のいずれか

第1章の“ SIGNAL MANAGEMENT ”は，信号について記述しています．

LINではしばしば信号（signal）ということばが登場します．この場合，信号は「LINフレームによってやりとりするデータ」のことを言います．各信号は一つの送信者によって送信され，ゼロ個以上のノードがこれを受信します．

信号の型は，スカラ値かバイト配列のいずれかと規定されています．スカラ値は1～16ビットの長さとされ，実際にAPIではbool（1ビット），u8（符号なし8ビット），u16（符号なし16ビット）の3種類が扱えるようになっています．

信号の一貫性についても定められており，たとえ信号の一部だけが変更された場合でも，データとしてすべて転送する必要があるとしています．つまり，信号のどこが変更されたのかを受信ノードのアプリケーションが判断しなければならないわけです．

実際の信号の並べかたは，LSB（least significant bit）から，配列は要素ゼロから昇順と規定されています．

● 各バイトの時間間隔が“負”になってはならない

第2章の“ FLAME TRANSFAR ”は，フレームの構造と転送シーケンスについて記述しています．

本章の冒頭の説明に重要なポイントが一つあります．各バイト間（バイト・スペース），ヘッダとレスポンスの間（レスポンス・スペース），各フレーム間（インター・フレーム・スペース）にはスペース，すなわちある時間間隔が挿入される可能性があります（**図7**）．ここでは，それらについて「負であってはならない」と記述されています．

では，実際に時間間隔が負になるようなことがあるのでしょうか．通常，バイト・スペースは同一タスクの連続送信で実現するので，ゼロ未満になることはまずありません．可能性があるのはレスポンス・スペースとインター・フレーム・スペースでしょう．

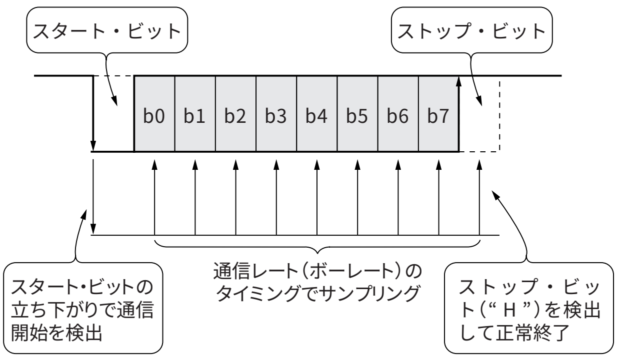


図8　UARTフォーマット
UARTによる1バイト送受信のもっとも一般的なフォーマットを示す．スタート・ビットと呼ばれる1ビット幅のロー・パルスでデータ送信が開始される．

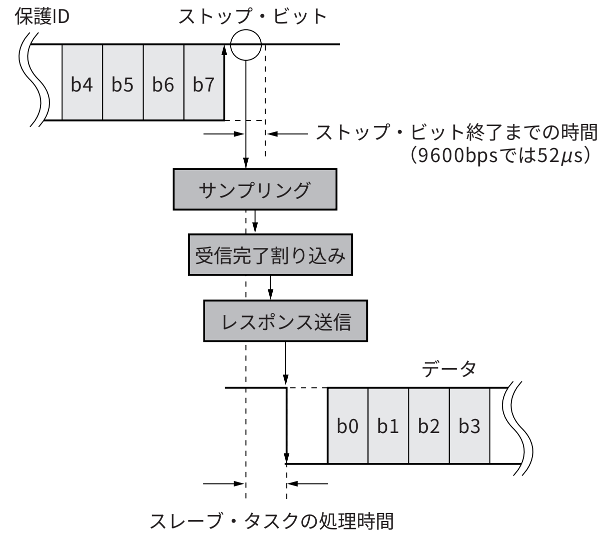


図9　レスポンス・スペース
保護IDのストップ・ビットが終了する前にスレーブ・タスクがレスポンスを返してしまう場合，レスポンス・スペースは負の値となり，仕様から逸脱する．

図8にUARTによる1バイト送受信のもっとも一般的なフォーマットを示します．スタート・ビットと呼ばれる1ビット幅のロー・パルスでデータ送信が開始されます．これに続いてデータをLSBから順番に8ビット送信し，最後にストップ・ビットと呼ばれる1ビット幅の“H”レベルで完了します．受信側はスタート・ビットの立ち下がりで通信開始を感知し，通信レートに従ってあてずっぽうに通信ラインをサンプリングしていきます．このように，送信側に歩調を合わせながら通信する方法を調歩同期と言います．通常，サンプリングはデータ・ビットの中央をねらって実施します．最後に，ストップ・ビットであるはずの位置でサンプリングを行い，“H”レベルであれば正常終了します．

次に，レスポンス・スペース，すなわちスレーブ・タスクが保護IDを受信してレスポンスの送信を開始する部分について検討してみます（**図9**）．

保護IDの受信完了は，ストップ・ビットの中央でサンプリングされます．ここから実際に保護IDのストップ・ビットが終了するまでは，例えば9,600bpsの場合で52 μs（＝1÷9600÷2）必要です．一方，スレーブ・タスクはストップ・ビットのサンプリングから受信完了割り込みを受け，データ（レスポンス）の送信処理へと移行します．この処理が高速である場合，例えば9,600bpsでは52 μs以下でレスポンスの送信を開始してしまう可能性があります．このとき，レスポンス・スペースは負の値となり，仕様から逸脱してしまいます．

したがって，レスポンス送信処理はレスポンス・スペースに対して少なくとも0s（秒）であることを保証する必要があります．レスポンス受信処理は，レスポンス・スペースがなくても（0であっても）受信できなければなりません．

インター・フレーム・スペースについても，同じような注意が必要です．

次号へつづく

参考・引用*文献

(1) * LIN Consortium ; LIN Specification Package Revision 2.0, Sep 2003.

たち・のぶゆき
NECマイクロシステム(株)

<筆者プロフィール>
舘 伸幸．組み込み系ソフトウェア・エンジニア．1983年入社以来，デバイス会社でソフトウェアひとすじ．